1. **Write a java program to reverse String without reverse method?**

```java
public class StringReverse {

    public static void main(String[] args) {
        String str = "abcd";
        char[] charArray = str.toCharArray();
        String reverse = "";

        for(int i = charArray.length;i>0;--i) {
            reverse = reverse + charArray[i-1];
        }

        System.out.println(reverse); //dcba
    }

}
```

2. **Write a java program to check given Strings anagram or not?**

```java
public class StringAnagram {

    public static void main(String[] args) {
        String word = "java2blog";
        String anagram = "aj2vabgol";

        boolean isAnagram = isDataAnagram(word, anagram);
        System.out.println(isAnagram); //true
    }

    private static boolean isDataAnagram(String word, String anagram) {
        boolean value = false;
        if (word.length() != anagram.length()) {
            return false;
        } else {
            for (int i = 0; i < word.length(); i++) {
                for (int j = 0; j < anagram.length(); j++) {
                    if (word.charAt(i) == anagram.charAt(j)) {
                        anagram = anagram.substring(0, j) + anagram.substring(j + 1);
                        System.out.println(anagram); //a2vabgol 2vabgol 2abgol 2bgol bgol gol go g

                        break;
                    }
                }
            }

            if (anagram.length() == 0) {
                value = true;
            } else {
                value = false;
            }
        }
        return value;
    }
```

3. **Write a Java program to check unique characters in String?**

```java
public class StringUniqueChars {

    public static void main(String[] args) {
        //Unique means there should not be duplicate characters
        String s1 = "Sunny";
        String s2 = "noise";

        System.out.println(identifyUniqueString(s1)); //false
        System.out.println(identifyUniqueString(s2)); //true
    }

    private static boolean identifyUniqueString(String input) {
        for(int i=0; i<input.length();i++) {
            for(int j= i+1; j<input.length();j++) {
                if(input.charAt(i)==input.charAt(j)) {
                    return false;
                }
            }
        }
        return true;
    }

}
```

4. Write a Java program to find out the duplicate characters in String?

```java
public class DuplicateCharsCountInString {

    public static void main(String[] args) {
        String str = "abcdabcd";

        int count=0;
        for (int i = 0; i < str.length(); i++) {
            count =1;
            for (int j = i + 1; j < str.length(); j++) {
                if (str.charAt(i) == str.charAt(j)) {
                    count++;
                    str = str.substring(0, j) + str.substring(j + 1);
                }
            }

            if (count > 1) {
                System.out.println(str.charAt(i) + " found " + count + " times"); //a found 2 times b found 2 times c found 2 times d found 2 times

            }
        }
    }

}
```

5. Write a program to remove the duplicate characters in String?

```java
public class RemoveDuplicatesInString {

    public static void main(String[] args) {
        String input ="abcdabcd";

        String unqueStr ="";
        for(int i=0;i<input.length();i++) {
            boolean duplicateFound= false;
            for(int j=i+1;j<input.length();j++) {
                if(input.charAt(i)== input.charAt(j)) {
                    duplicateFound = true;
                    break;
                }
            }

            if(!duplicateFound) {
                unqueStr = unqueStr+input.charAt(i);
            }
        }

        System.out.println(unqueStr); //abcd

    }

}
```

6. **Write a program to print all permutations of String in java?**

```java
public class StringPermutations {

    public static void main(String[] args) {
        String input ="abc";
        printPermutations(input,"");
    }

    private static void printPermutations(String input,String ans) {
        if (input.length() == 0) {
            System.out.print(ans + " "); //abc acb bac bca cab cba
            return;
        }

        for(int i=0;i<input.length();i++) {
            String data = input.substring(0,i)+input.substring(i+1);
            printPermutations(data,ans+input.charAt(i));
        }

    }

}
```

7. **Write a program to find the first non-repeated character from String?**

```
public class FirstNonRepeatedCharInString {

    public static void main(String[] args) {
        String str = "SuSny";

        for (int i = 0; i < str.length(); i++) {
            boolean unique = true;
            for (int j = i+1; j < str.length(); j++) {
                if (str.charAt(i) == str.charAt(j)) {
                    unique = false;
                    break;
                }
            }
            if (unique) {
                System.out.println(str.charAt(i));//u
                break;
            }
        }
    }

}
```

8. Write a program to check given string is palindrome or not?

First Approach:

```
public class PalindromExample {

    public static void main(String[] args) {
        String str = "malayalam";

        boolean palindrom = true;

         //Iterate the string forward and backward, compare one character at a time till middle of the string is reached
        for(int i=0;i < str.length()/2;i++) {
            if(str.charAt(i)!= str.charAt(str.length()-i-1)) {
                palindrom=false;
                break;
            }
        }

        if(palindrom) {
            System.out.println("Its Palindrom"); //Its Palindrom
        } else {
            System.out.println("It's not palindrom");
        }
    }

}
```

Second Approach

```java
public class PalindromExample {

    public static void main(String[] args) {
        String str = "malayalam";

        boolean palindrom = true;

        String reverse ="";
        for(int j=str.length();j>0;--j) {
            reverse = reverse + str.charAt(j-1);
        }

        if(str.equalsIgnoreCase(reverse)) {
            System.out.println("Its Palindrom"); //Its Palindrom
        } else {
            System.out.println("It's not palindrom");
        }
```

9. Write java Program to Find Smallest and Largest Element in an Array?

```java
public class SmallestAndLargeArray {

    public static void main(String[] args) {
        int[] intArr = { 9, 3, 2, 1 };

        int smallest = intArr[0];
        int largest = intArr[0];

        for (int i = 1; i < intArr.length; i++) {
            if (intArr[i] > largest) {
                largest = intArr[i];
            } else if (intArr[i] < smallest) {
                smallest = intArr[i];

            }
        }
        System.out.println("Smallest Number is : " + smallest); //Smallest Number is : 1
        System.out.println("Largest Number is : " + largest); //Largest Number is : 9
    }
}
```

10. Find missing number in the array?

```java
public class MissingNumberInArray {

    public static void main(String[] args) {
        int[] arr={7,5,6,1,4,2};

        System.out.println(getMissingNumber(arr)); //3
    }

    private static int getMissingNumber(int[] arr) {
        int n = arr.length+1;
        int sum = n*(n+1)/2;
        int remainSum=0;

        for(int i=0;i<arr.length;i++) {
            remainSum += arr[i];
        }

        System.out.println("Sum is " + sum + " remain Sum " +remainSum);//Sum is 28 remain Sum 25
        return  sum-remainSum;
    }
```

11. Find second largest number in an array?

```java
public class SecondLargestInArray {

    public static void main(String[] args) {
        int[] arr = { 7, 5, 6, 1, 4, 2 };

        int temp;
        for(int i=0;i<arr.length;i++) {
            for(int j =i+1;j<arr.length;j++) {
                if(arr[i]>arr[j]) {
                    temp = arr[i];
                    arr[i]=arr[j];
                    arr[j]= temp;
                }
            }
        }

        System.out.println("Second Largest " + arr[arr.length-2]);//Second Largest 6
        System.out.println("Third Largest " + arr[arr.length-3]);//Third Largest 5

    }

}
```

12. Write a logic to remove duplicates from Array?

```java
public class RemoveDuplicatesFromArrays {

    public static void main(String[] args) {

        int[] arrNumbers = { 1, 2, 3, 3, 4, 5, 5, 6, 7, 8,9 };
        int len = arrNumbers.length;
        len = removeDuplicate(arrNumbers, len);
        System.out.println(len); //9 based on length numbers will be printed
        for (int a = 0; a < len; a++) {
            System.out.print(arrNumbers[a] + " "); //1 2 3 4 5 6 7 8 9
        }

    }

    private static int removeDuplicate(int[] arrNumbers, int num) {
        int b = 0;
        for (int a = 0; a < num - 1; a++) {
            if (arrNumbers[a] != arrNumbers[a + 1]) {
                arrNumbers[b++] = arrNumbers[a];
            }
        }
        arrNumbers[b++] = arrNumbers[num - 1];
        return b;
    }

}
```

13. Write a logic to find common elements from two arrays?

```java
public class CommonElementsInArray {

    public static void main(String[] args) {
        int a[] = { 1, 2, 4, 5 };
        int b[] = { 1, 3, 5 };

        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < b.length; j++) {
                if(a[i] == b[j]) {
                    System.out.print(a[i] + " "); //1 5
                }
            }
        }

    }

}
```

14. Write a program to print the array in ascending and descending order?

```
public class ArrayOrder {

    public static void main(String[] args) {
        int[] arrNumbers = { 1, 8, 3, 4, 9, 5, 6, 7, 2 };

        int arrLength = arrNumbers.length;

        for(int i=0;i<arrLength;i++) {
            for(int j=i+1;j<arrLength;j++) {
                if(arrNumbers[i]>arrNumbers[j]) {
                    int temp = arrNumbers[i];
                    arrNumbers[i] = arrNumbers[j];
                    arrNumbers[j] = temp;
                }
            }
        }

        //Ascending
        for(int ascending: arrNumbers) {
            System.out.print(ascending + " ");//1 2 3 4 5 6 7 8 9
        }

        //Descending
        for(int desc=arrLength;desc>0;--desc) {
            System.out.print(desc + " "); //9 8 7 6 5 4 3 2 1
        }

    }

}
```

15. Write a program to build Single Linked List?

```
class Node {
    int data;
    Node next;

    public void displayNodeData() {
        System.out.println("{ " + data + " } ");
    }
}

public class SingleLinkedList {
    private Node head;
    public void insert(int data) {
        Node newNode = new Node();
        newNode.data = data;
        newNode.next = head;
        head = newNode;
    }
    public Node delete() {
        Node temp = head;
        head = head.next;
        return temp;
    }
    public void printLinkedList() {
        System.out.println("Printing LinkedList (head --> last) ");
        Node current = head;
        while (current != null) {
            current.displayNodeData();
            current = current.next;
        }
        System.out.println();
    }

    public static void main(String[] args) {
        SingleLinkedList singleLinkedList = new SingleLinkedList();
        singleLinkedList.insert(2);
        singleLinkedList.insert(5);
        singleLinkedList.printLinkedList();

        singleLinkedList.delete();
        singleLinkedList.printLinkedList();
    }
}
```

16. Write a program for Double Linked List?

```
class DoubleNode {
    int data;
    DoubleNode previous;
    DoubleNode next;

    public DoubleNode(int data) {
        this.data = data;
    }
}

public class DoubleLinkedList {
    DoubleNode head = null;
    DoubleNode tail = null;

    public void insert(int data) {
        DoubleNode newNode = new DoubleNode(data);
        if (head == null) {
            head = tail = newNode;
            head.previous = null;
            tail.next = null;
        } else {
            tail.next = newNode;
            newNode.previous = tail;
            tail = newNode;
            tail.next = null;
        }
    }

    public void display() {
        DoubleNode current = head;
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next;
        }
    }

    public static void main(String[] args) {
        DoubleLinkedList doubleLinkedList = new DoubleLinkedList();
        doubleLinkedList.insert(25);
        doubleLinkedList.display();
    }
```

17. Write a program for Stack?

```
public class MyStack {

    private int maxSize;
    private long[] stackArray;
    private int top;

    public MyStack(int s) {
        maxSize = s;
        stackArray = new long[maxSize];
        top = -1;
    }

    public void push(long j) {
        stackArray[++top] = j;
    }

    public long pop() {
        return stackArray[top--];
    }

      public boolean isEmpty() {
          return (top == -1);
       }

    public static void main(String[] args) {

        MyStack theStack = new MyStack(10);
        theStack.push(90);
        theStack.push(60);
        theStack.push(30);

         while (!theStack.isEmpty()) {
             long value = theStack.pop();
             System.out.print(value);
             System.out.print(" ");
          }
          System.out.println("");

    }

}
```

18. Write a program to build Queue?

```
public class QueueExample {
    int capacity;
    int queueArr[];
    int front;
    int rear;
    int currentSize = 0;

    public QueueExample(int sizeOfQueue) {
        this.capacity = sizeOfQueue;
        front = 0;
        rear = -1;
        queueArr = new int[this.capacity];
    }
    public void enqueue(int data) {
        if (isFull()) {
            System.out.println("Queue is full!! Can not add more elements");
        } else {
            rear++;
            if (rear == capacity - 1) {
                rear = 0;
            }
            queueArr[rear] = data;
            currentSize++;
            System.out.println(data + " added to the queue");
        }
    }
    public void dequeue() {
        if (isEmpty()) {
            System.out.println("Queue is empty!! Can not dequeue element");
        } else {
            front++;
            if (front == capacity - 1) {
                System.out.println(queueArr[front - 1] + " removed from the queue");
                front = 0;
            } else {
                System.out.println(queueArr[front - 1] + " removed from the queue");
            }
            currentSize--;
        }
    }

  public boolean isFull() {
      if (currentSize == capacity) {
          return true;
      }
      return false;
  }

  public boolean isEmpty() {

      if (currentSize == 0) {
          return true;
      }
      return false;
  }

  public static void main(String[] args) {
      QueueExample queue = new QueueExample(5);
      queue.enqueue(6);
      queue.dequeue();
  }
```